Claims:

1. A method for managing memory in a system including one or more memory modules, wherein at least one of the one or more memory modules includes a decompression engine, the method comprising:

storing compressed data on the one or more memory modules;

a device initiating a read of requested data from the one or more memory modules, wherein the requested data comprises compressed requested data stored on the one or more memory modules in a compressed format;

the one or more memory modules decompressing the compressed requested data to produce uncompressed requested data using parallel decompression, wherein said decompressing the compressed requested data comprises:

examining a plurality of tokens from the compressed requested data in parallel in a current decompression cycle, wherein each of the plurality of tokens describes one or more symbols in the uncompressed requested data;

generating a plurality of selects in parallel in response to said examining the plurality of tokens in parallel, wherein each of the plurality of selects points to a symbol in a combined history window; and

generating the uncompressed requested data comprising the plurality of symbols using the plurality of selects; and

providing the uncompressed requested data to the device.

2. The method of claim 1, further comprising:

storing the uncompressed plurality of symbols from the current decompression cycle in the combined history window.

3. The method of claim 1, wherein said examining the plurality of tokens includes generating, for each token, size and count information and at least one of a data byte or index information;

wherein said generating the plurality of selects in parallel uses the size and count information and at least one of the data byte or index information for each of the plurality of tokens;

wherein a size for a token defines the number of bits comprising the token; and

wherein a count for a token defines the number of symbols in the uncompressed data described by the token.

4. The method of claim 1, wherein the combined history window includes an uncompressed plurality of symbols from one or more previous decompression cycles and data bytes from the current decompression cycle, wherein said generating the plurality of selects in parallel comprises:

generating a first select to point to a data byte in the combined history window in response to a first token indicating that uncompressed data represented by the first token is the data byte; and

generating a second select to point to a first symbol in the combined history window in response to a second token indicating that uncompressed data represented by the second token includes the first symbol in the combined history window.

5. The method of claim 4, wherein the uncompressed data represented by the second token includes one or more symbols following the first symbol in the combined history window, wherein selects are generated to point to each of the one or more symbols in the combined history window comprising the uncompressed data represented by the second token.

6. The method of claim 1, wherein the combined history window includes an uncompressed plurality of symbols from one or more previous decompression cycles and data bytes from the current decompression cycle, wherein said generating the plurality of selects in parallel comprises:

generating a first select to point to a first symbol being decompressed from a first token in the current decompression cycle, wherein the first select is generated in response

to a second token indicating that uncompressed data represented by the second token includes the first symbol, and wherein the first symbol is not in the combined history window.

7. The method of claim 6, further comprising resolving the first select to point to one of a symbol in the current combined history window or a data byte in the current combined history window.

8. The method of claim 7, further comprising copying a second select being generated for the first token to the first select, wherein the second select points to one of a symbol in the combined history window or a data byte in the combined history window.

9. The method of claim 1, wherein the combined history window includes an uncompressed plurality of symbols from one or more previous decompression cycles, and wherein said storing the uncompressed plurality of symbols from the current decompression cycle in the combined history window includes removing from the combined history window at least a portion of the uncompressed plurality of symbols from the one or more previous decompression cycles.

10. The method of claim 1, wherein said examining the plurality of tokens in parallel comprises:

extracting a portion of the compressed data as an input data, wherein the input data includes the plurality of tokens;

extracting one or more tokens from the input data; and

generating, for each token, size and count information and at least one of a data byte or index information.

11. The method of claim 10, wherein a plurality of decoders are operable to examine the plurality of tokens in parallel in the current decompression cycle, wherein said extracting the one or more tokens comprises:

a) determining if there are more tokens to be uncompressed in the input data;

b) determining if there is a decoder available in the plurality of decoders;

c) extracting a token from the input data to be a current token in response to determining that a decoder is available;

d) determining the number of uncompressed symbols to be generated by the current token;

e) assigning the current token to the available decoder; and

f) repeating a) through e) until a terminating condition is encountered.

12. The method of claim 11, wherein the terminating condition is encountered when:

step a) determines there are no more tokens in the input data; or

step b) determines there are no more decoders available in the plurality of decoders; or

step d) determines that a total number of uncompressed symbols to be generated from the plurality of tokens to be examined in parallel has met or exceeded a maximum output width.

13. The method of claim 11, wherein said extracting the one or more tokens further comprises determining a size of each of the one or more tokens.

14. The method of claim 11, wherein said extracting the one or more tokens further comprises:

determining if a token in the input data is a complete token or an incomplete token, wherein the token is extracted from the input data to be the current token in response to determining the token is a complete token;

wherein the terminating condition is encountered in response to determining the token is an incomplete token.

15. The method of claim 1, wherein said receiving the compressed data, said examining a plurality of tokens, said generating a plurality of selects, and said generating the uncompressed data comprising the plurality of symbols are performed substantially concurrently in a pipelined fashion.

16. The method of claim 1, wherein at least one of the one or more memory modules further includes a compression engine, the method further comprising:

a device initiating a write of uncompressed data to the one or more memory modules, wherein the uncompressed data is in an uncompressed format;

the one or more memory modules receiving the uncompressed data from the device; and

the one or more memory modules compressing the uncompressed data to produce compressed data, wherein the uncompressed data comprises a plurality of symbols, wherein the one or more memory modules compressing the uncompressed data comprises:

a) comparing a plurality of symbols from the uncompressed data with each entry in a history table in a parallel fashion, wherein said comparing produces compare results;

wherein the history table comprises entries, wherein each entry comprises at least one symbol, and wherein the method maintains a current count of prior matches which occurred when previous symbols were compared with entries in the history table;

b) determining match information for each of said plurality of symbols based on the current count and the compare results; and

c) outputting compressed data in response to the match information; and

storing the compressed data on the one or more memory modules.

17. A method for managing memory in a system including one or more memory modules, wherein at least one of the one or more memory modules includes a compression engine, the method comprising:

a device initiating a write of uncompressed data to the one or more memory modules, wherein the uncompressed data is in an uncompressed format;

the one or more memory modules receiving the uncompressed data from the device; and

the one or more memory modules compressing the uncompressed data to produce compressed data, wherein the uncompressed data comprises a plurality of symbols, wherein the one or more memory modules compressing the uncompressed data comprises:

a) comparing a plurality of symbols from the uncompressed data with each entry in a history table in a parallel fashion, wherein said comparing produces compare results;

wherein the history table comprises entries, wherein each entry comprises at least one symbol, and wherein the method maintains a current count of prior matches which occurred when previous symbols were compared with entries in the history table;

b) determining match information for each of said plurality of symbols based on the current count and the compare results; and

c) outputting compressed data in response to the match information; and

storing the compressed data on the one or more memory modules.

18. The method of claim 17, wherein said outputting compressed data includes:

outputting a count value and an entry pointer for a contiguous match, wherein the entry pointer points to the entry in the history table which produced the contiguous match, wherein the count value indicates a number of matching symbols in the contiguous match.

19. The method of claim 25, wherein said outputting the count value includes encoding a value representing the count value; wherein more often occurring counts are encoded with fewer bits than less often occurring counts.

20. The method of claim 17, wherein said outputting compressed data further includes:

for non-matching symbols which do not match any entry in the history table, outputting the non-matching symbols.

21. The method of claim 17, further comprising:

d) repeating steps a) – c) one or more times until no more data is available; and

e) when no more data is available, if the current count is non-zero, outputting compressed data for the remaining match in the history table.

22. The method of claim 21, wherein said determining match information includes determining zero or more matches of said plurality of symbols with each entry in the history table.

23. The method of claim 17, wherein the method further maintains a count flag for each entry in the history table;

wherein said determining determines match information for each of said plurality of symbols based on the current count, the count flags, and the compare results.

24. The method of claim 23, wherein said determining match information includes:

resetting the count and count flags if the compare results indicate a contiguous match did not match one of the plurality of symbols.

25. The method of claim 23, wherein the count and count flags for all entries are reset based on the number of the plurality of symbols that did not match in the contiguous match.

26. The method of claim 17, wherein said determining match information includes:

updating the current count according to the compare results.

27. The method of claim 17, wherein said determining match information includes:

determining a contiguous match based on the current count and the compare results;

determining if the contiguous match has stopped matching;

if the contiguous match has stopped matching, then:

updating the current count according to the compare results ; and

wherein said outputting compressed data includes outputting compressed data corresponding to the contiguous match.

28. The method of claim 27, wherein said outputting compressed data corresponding to the contiguous match comprises outputting a count value and an entry pointer, wherein the entry pointer points to the entry in the history table which produced the contiguous match, wherein the count value indicates a number of matching symbols in the contiguous match.

29. The method of claim 17, wherein the plurality of symbols includes a first symbol, a last symbol, and one or more middle symbols;

wherein said determining match information includes:

if at least one contiguous match occurs with one or more respective contiguous middle symbols, and the one or more respective contiguous middle symbols are not involved in a match with either the symbol before or after the respective contiguous middle symbols, then:

selecting the one or more largest non-overlapping contiguous matches involving the middle symbols;

wherein said outputting compressed data includes outputting compressed data for each of the selected matches involving the middle symbols.

30. The method of claim 17,

wherein the method further maintains a count flag for each entry in the history table;

wherein said determining determines match information for each of said plurality of symbols based on the current count, the count flags, and the compare results;

wherein said determining match information and said outputting compressed data in response to the match information comprises:

determining zero or more matches of said plurality of symbols with each entry in the history table;

examining the compare results for each entry;

for non-matching symbols which do not match any entry in the history table, outputting the non-matching symbols;

if any entry stopped matching, examining the current count, the count flags, and the compare results for every entry;

determining the contiguous match based on the current count and the compare results;

determining if the contiguous match has stopped matching;

if the contiguous match has stopped matching, then:

outputting a count value and an entry pointer, wherein the entry pointer points to the entry in the history table which produced the contiguous match, wherein the count value indicates a number of matching symbols in the contiguous match; and

updating the current count according to the compare results;

the method further comprising:

e) repeating steps a) – c) one or more times until no more data is available; and

f) when no more data is available, if the current count is non-zero, outputting a count value and an entry pointer for the remaining match in the history table.

31. The method of claim 30, wherein the plurality of symbols includes a first symbol, a last symbol, and one or more middle symbols;

wherein, if the contiguous match has stopped matching, then the method further comprises:

if at least one contiguous match occurs with one or more respective contiguous middle symbols, and the one or more respective contiguous middle symbols are not involved in a match with either the symbol before or after the respective contiguous middle symbols, then:

selecting the largest non-overlapping contiguous matches involving the middle symbols; and

outputting a count value and an entry pointer for each of the selected matches involving the middle symbols.

32. The method of claim 17, wherein the plurality of symbols comprise a power of 2 number of symbols.

33. The method of claim 17, further comprising:

generating a header for the compressed data; and

storing the header on the solid state memory;

wherein the header includes information for decompressing the compressed data.

34. A memory module, comprising:

one or more memory devices for storing data;

a compression/decompression engine for compressing uncompressed data, the uncompressed data having a plurality of symbols, and for decompressing compressed data, wherein the compressed data comprises a compressed representation of uncompressed data;

wherein the compression/decompression engine is operable to:

receive a first uncompressed data; and

compress the first uncompressed data to produce a first compressed data.

35. The memory module of claim 34,

wherein, in said compressing the first uncompressed data, the compression/decompression engine is further operable to:

compare a plurality of symbols from the first uncompressed data with each entry in a history table in a parallel fashion, wherein said comparing produces compare results;

wherein the history table comprises entries, wherein each entry comprises at least one symbol, and wherein the compression/decompression engine maintains a current count of prior matches which occurred when previous symbols were compared with entries in the history table;

determine match information for each of said plurality of symbols based on the current count and the compare results; and

output compressed data in response to the match information.

36. The memory module of claim 34, wherein the compression/decompression engine is further operable to:

store the first compressed data on the memory module.

37. The memory module of claim 34, wherein the compression/decompression engine is further operable to:

receive a second compressed data; and

decompress the second compressed data to produce a second uncompressed data in response to said receiving the second compressed data.

38. The memory module of claim 37,

wherein the second compressed data comprises tokens each describing one or more uncompressed symbols; and

wherein, in said decompressing the second compressed data, the compression/decompression engine is further operable to:

examine a plurality of tokens from the second compressed data in parallel in a current decompression cycle;

generate a plurality of selects in parallel in response to examining the plurality of tokens in parallel, wherein each of the plurality of selects points to a symbol in a combined history window; and

generate uncompressed data comprising the plurality of symbols.

39. A memory module, comprising:

one or more memory devices for storing data;

a parallel compression engine for compressing data, wherein the parallel compression engine is operable to:

maintain a history table comprising entries, wherein each entry comprises at least one symbol;

receive uncompressed data, wherein the uncompressed data comprises a plurality of symbols;

compare a plurality of symbols with entries in the history table in a parallel fashion, wherein said comparing produces compare results;

determine match information for each of the plurality of symbols based on the compare results; and

output compressed data in response to the match information.

40. The memory module of claim 39, wherein the parallel compression engine is operable to output a count value and an entry pointer for a contiguous match, wherein the entry pointer points to the entry in the history table which produced the contiguous match, wherein the count value indicates a number of matching symbols in the contiguous match.

41. The memory module of claim 39, wherein the parallel compression engine is operable to determine zero or more matches of said plurality of symbols with each entry in the history table.

42. The memory module of claim 39, wherein the parallel compression engine is operable to maintain count information including a count of prior matches which occurred when previous symbols were compared with entries in the history table;

wherein the parallel compression engine is operable to determine the match information for each of said plurality of symbols based on the count information and the compare results.

43. A memory module, comprising:

one or more memory devices for storing data;

a parallel compression engine for compressing data, wherein the parallel compression engine comprises:

an input for receiving uncompressed data, wherein the uncompressed data comprises a plurality of symbols;

a history table comprising entries, wherein each entry comprises at least one symbol;

a plurality of comparators for comparing a plurality of symbols with entries in the history table in a parallel fashion, wherein the plurality of comparators produce compare results;

match information logic coupled to the plurality of comparators for determining match information for each of said plurality of symbols based on the compare results; and

an output coupled to the match information logic for outputting compressed data in response to the match information.

44. The memory module of claim 43, wherein said compressed data includes a count value and an entry pointer for a contiguous match, wherein the entry pointer points to the entry in the history table which produced the contiguous match, wherein the count value indicates a number of matching symbols in the contiguous match.

45. The memory module of claim 43, wherein the match information logic is operable to determine zero or more matches of said plurality of symbols with each entry in the history table.

46. The memory module of claim 43, further comprising:

a memory which maintains count information including a count of prior matches which occurred when previous symbols were compared with entries in the history table.

47. A memory module, comprising:

one or more memory devices for storing data;

a parallel decompression engine for decompressing compressed data, wherein the compressed data comprises a compressed representation of uncompressed data, the uncompressed data having a plurality of symbols, wherein the parallel decompression engine is operable to:

receive the compressed data, wherein the compressed data comprises tokens each describing one or more uncompressed symbols;

examine a plurality of tokens from the compressed data in parallel in a current decompression cycle;

generate a plurality of selects in parallel in response to examining the plurality of tokens in parallel, wherein each of the plurality of selects points to a symbol in a combined history window;

generate uncompressed data comprising the plurality of symbols.

48. The memory module of claim 47, wherein the parallel decompression engine is further operable to:

store the uncompressed plurality of symbols from the current decompression cycle in the combined history window.

49. The memory module of claim 47, wherein said examining the plurality of tokens includes generating, for each token, size and count information and at least one of a data byte or index information; and

wherein said generating the plurality of selects in parallel uses the size and count information and at least one of the data byte or index information for each of the plurality of tokens.

50. The memory module of claim 47, wherein the combined history window includes an uncompressed plurality of symbols from one or more previous decompression cycles and data bytes from the current decompression cycle, wherein said generating the plurality of selects in parallel comprises:

generating a first select to point to a data byte in the combined history window in response to a first token indicating that uncompressed data represented by the first token is the data byte; and

generating a second select to point to a first symbol in the combined history window in response to a second token indicating that uncompressed data represented by the second token includes the first symbol in the combined history window.

51. The memory module of claim 47, wherein the combined history window includes an uncompressed plurality of symbols from one or more previous decompression cycles and data bytes from the current decompression cycle, wherein said generating the plurality of selects in parallel comprises:

generating a first select to point to a first symbol being decompressed from a first token in the current decompression cycle, wherein the first select is generated in response to a second token indicating that uncompressed data represented by the second token includes the first symbol, and wherein the first symbol is not in the combined history window.

52. The memory module of claim 47, further comprising copying a second select being generated for the first token to the first select, wherein the second select points

to one of a symbol in the combined history window or a data byte in the combined history window.

53. A memory module, comprising:

one or more memory devices for storing data;

a parallel decompression engine for decompressing compressed data, wherein the compressed data comprises a compressed representation of uncompressed data, the uncompressed data having a plurality of symbols, wherein the parallel decompression engine comprises:

an input for receiving the compressed data, wherein the compressed data comprises tokens each describing one or more uncompressed symbols;

a combined history window comprising a plurality of uncompressed symbols;

token examination logic operable to examine a plurality of tokens from the compressed data in parallel in a current decompression cycle;

preliminary select generation logic operable to generate a plurality of preliminary selects in parallel in response to the token examination logic examining the plurality of tokens in parallel, wherein each of the plurality of preliminary selects points to an uncompressed symbol in the combined history window or to another of the preliminary selects in the current decompression cycle;

final select generation logic operable to generate a plurality of final selects in parallel in response to the preliminary select generation logic generating the plurality of preliminary selects, wherein each of the plurality of final selects points to an uncompressed symbol in the combined history window;

uncompressed data output logic operable to:

generate the uncompressed data comprising the uncompressed symbols pointed to by the plurality of final selects generated by the second stage; and

store the uncompressed symbols from the current decompression cycle in the combined history window; and

an output coupled to the uncompressed data output logic for outputting the plurality of uncompressed symbols in the uncompressed data in response to the decompression of the plurality of tokens.

54. The memory module of claim 53, wherein the token examination logic includes a plurality of decoders, wherein the decoders are operable to each receive one token from the plurality of tokens in the current decompression cycle, and wherein the decoders are operable to:

generate, for each token, size and count information and at least one of a data byte or index information;

wherein the preliminary select generation logic uses the size and count information and at least one of the data byte or index information generated by the decoders to generate the plurality of preliminary selects in parallel.

55. A computer system including a memory module having an embedded parallel compression/decompression engine, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications, wherein the system memory includes one or more memory modules, wherein at least one of the one or more memory modules includes the parallel compression/decompression engine for compressing data transferred to or from the system memory and for decompressing compressed data, wherein the compressed data comprises a compressed representation of uncompressed data, the uncompressed data having a plurality of symbols; and

a memory controller coupled to the system memory and the CPU, wherein the memory controller performs memory control functions for the system memory.

56. The computer system of claim 55,

wherein the parallel compression/decompression engine is operable to:

maintain a history table comprising entries, wherein each entry comprises at least one symbol;

receive a first uncompressed data, wherein the first uncompressed data comprises a plurality of symbols;

compare a plurality of symbols with entries in the history table in a parallel fashion, wherein said comparing produces compare results;

determine match information for each of the plurality of symbols based on the compare results; and

output a first compressed data in response to the match information.

57. The computer system of claim 55,

wherein the parallel compression/decompression engine is operable to:

receive a second compressed data, wherein the second compressed data comprises tokens each describing one or more uncompressed symbols;

examine a plurality of tokens from the second compressed data in parallel in a current decompression cycle;

generate a plurality of selects in parallel in response to examining the plurality of tokens in parallel, wherein each of the plurality of selects points to a symbol in a combined history window;

generate a second uncompressed data comprising the plurality of symbols; and

store the uncompressed plurality of symbols from the current decompression cycle in the combined history window.

58. A computer system including a memory module having an embedded parallel compression engine, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications, wherein the system memory includes one or more memory modules, wherein

at least one of the one or more memory modules includes the parallel compression engine for compressing data transferred to or from the system memory;

a memory controller coupled to the system memory and the CPU, wherein the memory controller performs memory control functions for the system memory;

wherein the parallel compression engine is operable to:

maintain a history table comprising entries, wherein each entry comprises at least one symbol;

receive uncompressed data, wherein the uncompressed data comprises a plurality of symbols;

compare a plurality of symbols with entries in the history table in a parallel fashion, wherein said comparing produces compare results;

determine match information for each of the plurality of symbols based on the compare results; and

output compressed data in response to the match information.

59. A computer system including a memory module having an embedded parallel compression engine, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications, wherein the system memory includes one or more memory modules, wherein at least one of the one or more memory modules includes the parallel compression engine for compressing data transferred to or from the system memory;

a memory controller coupled to the system memory and the CPU, wherein the memory controller performs memory control functions for the system memory;

wherein the parallel compression engine comprises:

an input for receiving uncompressed data, wherein the uncompressed data comprises a plurality of symbols;

a history table comprising entries, wherein each entry comprises at least one symbol;

a plurality of comparators for comparing a plurality of symbols with entries in the history table in a parallel fashion, wherein the plurality of comparators produce compare results;

match information logic coupled to the plurality of comparators for determining match information for each of said plurality of symbols based on the compare results; and

an output coupled to the match information logic for outputting compressed data in response to the match information.

60. A computer system including a memory module having an embedded parallel decompression engine for decompressing compressed data, wherein the compressed data comprises a compressed representation of uncompressed data, the uncompressed data having a plurality of symbols, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications, wherein the system memory includes one or more memory modules, wherein at least one of the one or more memory modules includes the parallel compression engine for compressing data transferred to or from the system memory;

a memory controller coupled to the system memory and the CPU, wherein the memory controller performs memory control functions for the system memory;

wherein the parallel decompression engine is operable to:

receive the compressed data, wherein the compressed data comprises tokens each describing one or more uncompressed symbols;

examine a plurality of tokens from the compressed data in parallel in a current decompression cycle;

generate a plurality of selects in parallel in response to examining the plurality of tokens in parallel, wherein each of the plurality of selects points to a symbol in a combined history window;

generate uncompressed data comprising the plurality of symbols; and

store the uncompressed plurality of symbols from the current decompression cycle in the combined history window.

61. A computer system including a memory module having an embedded parallel decompression engine for decompressing compressed data, wherein the compressed data comprises a compressed representation of uncompressed data, the uncompressed data having a plurality of symbols, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications, wherein the system memory includes one or more memory modules, wherein at least one of the one or more memory modules includes the parallel compression engine for compressing data transferred to or from the system memory;

a memory controller coupled to the system memory and the CPU, wherein the memory controller performs memory control functions for the system memory;

wherein the parallel decompression engine comprises:

an input for receiving the compressed data, wherein the compressed data comprises tokens each describing one or more uncompressed symbols;

a combined history window comprising a plurality of uncompressed symbols;

token examination logic operable to examine a plurality of tokens from the compressed data in parallel in a current decompression cycle;

preliminary select generation logic operable to generate a plurality of preliminary selects in parallel in response to the token examination logic examining the plurality of tokens in parallel, wherein each of the plurality of preliminary selects points to an uncompressed symbol in the combined history window or to another of the preliminary selects in the current decompression cycle;

final select generation logic operable to generate a plurality of final selects in parallel in response to the preliminary select generation logic generating the plurality of

preliminary selects, wherein each of the plurality of final selects points to an uncompressed symbol in the combined history window;

uncompressed data output logic operable to:

generate the uncompressed data comprising the uncompressed symbols pointed to by the plurality of final selects; and

store the uncompressed symbols from the current decompression cycle in the combined history window; and

an output coupled to the uncompressed data output logic for outputting the plurality of uncompressed symbols in the uncompressed data in response to the decompression of the plurality of tokens.